



# FSE

FOUNDATIONS OF  
SOFTWARE ENGINEERING

## Software Prototyping

Rapid software development to  
validate requirements

G51FSE

# Objectives

- \* To describe the use of prototypes in different types of development project
- \* To discuss evolutionary and throw-away prototyping
- \* To introduce three rapid prototyping techniques - high-level language development, database programming and component reuse
- \* To explain the need for user interface prototyping

# System prototyping

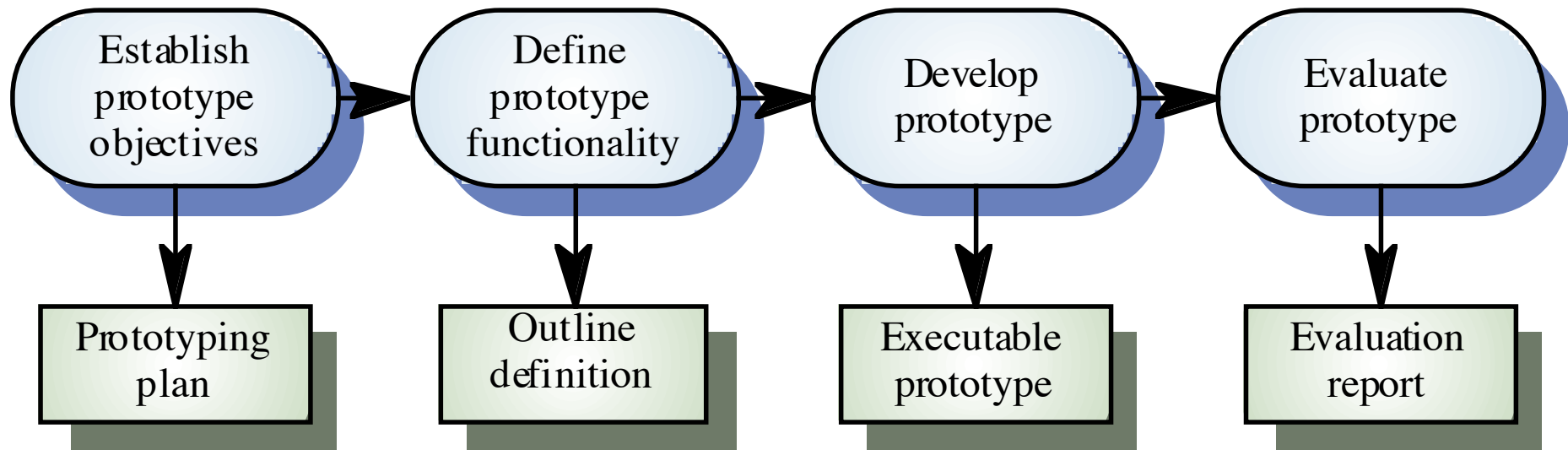
- \* Prototyping is the rapid development of a system
- \* In the past, the developed system was normally thought of as inferior in some way to the required system so further development was required
- \* Now, the boundary between prototyping and normal system development is blurred
- \* Many systems are developed using an evolutionary approach

# Why bother?

- \* The principal use is to help customers and developers understand the requirements for the system
  - \* Requirements *elicitation*: users can experiment with a prototype to see how the system supports their work
  - \* Requirements *validation*: The prototype can reveal errors and omissions in the requirements
- \* Prototyping can be considered as a **risk reduction activity** which reduces requirements risks

# Prototyping benefits

- \* Misunderstandings between software users and developers are exposed
- \* Missing services may be detected and confusing services may be identified
- \* A working system is available early in the process
- \* The prototype may serve as a basis for deriving a system specification
- \* The system can support user training and system testing



# Prototyping benefits

- \* Improved system usability
- \* Closer match to the system needed
- \* Improved design quality
- \* Improved maintainability
  - \* Fewer bugs and extensibility issues
- \* Reduced overall development effort

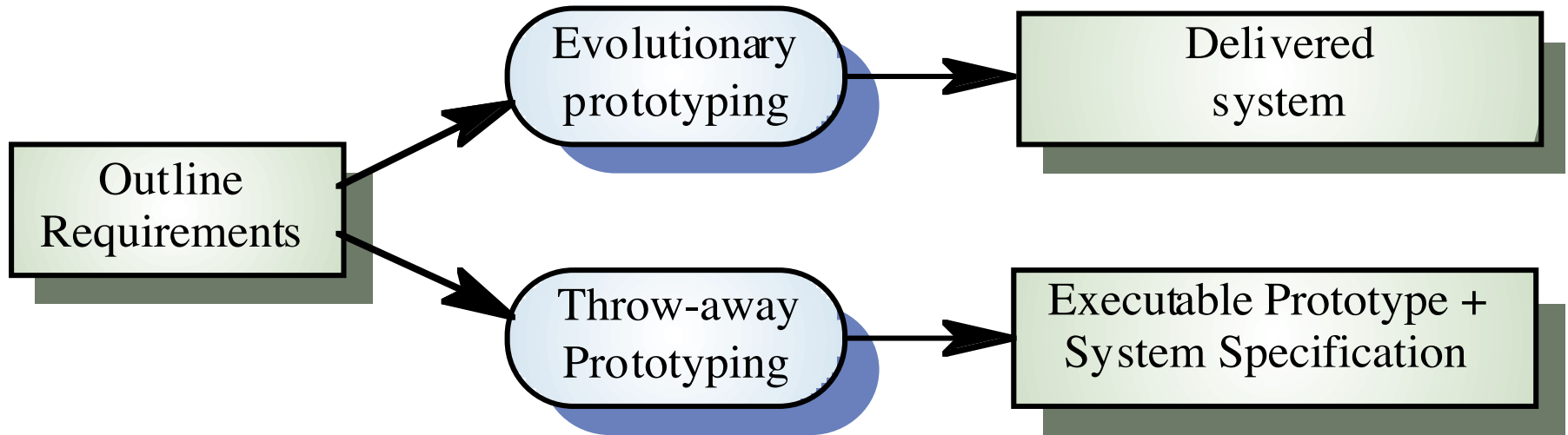
# Two different approaches

- \* Evolutionary prototyping:
  - \* an initial prototype is produced and refined through a number of stages to the final system
- \* Throw-away prototyping:
  - \* a practical implementation of the system is produced to help discover requirements problems and then discarded
  - \* the system is then developed using some other development process



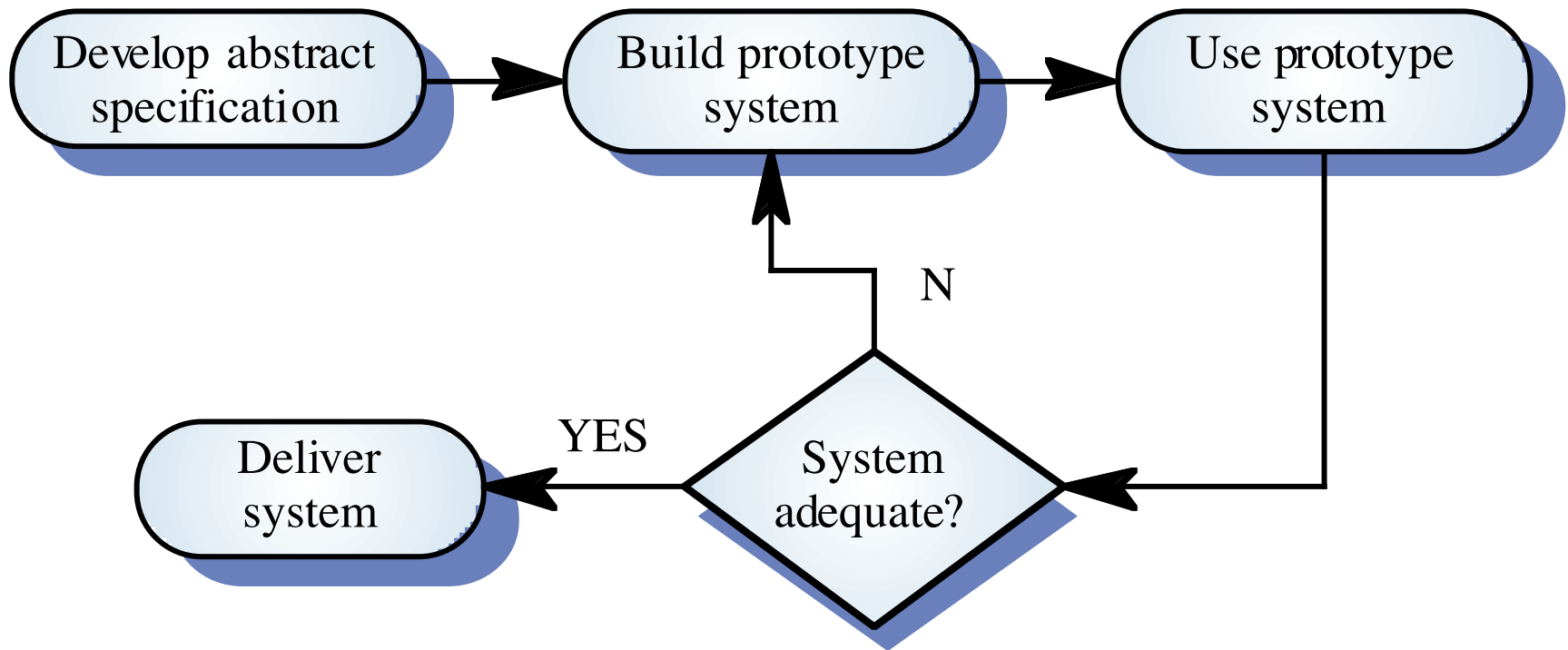
# Prototyping objectives

- \* The objective of *evolutionary prototyping* is to deliver a working system to end-users
  - \* The development starts with those requirements which are *best* understood.
- \* The objective of *throw-away prototyping* is to validate or derive the system requirements
  - \* The prototyping process starts with those requirements which are *poorly* understood



# Evolutionary prototyping

- \* Must be used for systems where the specification cannot be developed in advance e.g. AI systems and user interface systems
- \* Based on techniques which allow rapid system iterations
- \* Verification is impossible as there is no formal specification
  - \* Validation means demonstrating the adequacy of the system - does what it says on the tin



# Evolutionary prototyping advantages

- \* Accelerated delivery of the system
  - \* Rapid delivery and deployment are sometimes more important than functionality or long-term software maintainability
- \* User engagement with the system
  - \* Not only is the system more likely to meet user requirements, they are more likely to commit to the use of the system

# Evolutionary prototyping

- \* Specification, design and implementation are intertwined
- \* The system is developed as a series of increments that are delivered to the customer
- \* Techniques for rapid system development are used such as CASE tools and 4GLs
- \* User interfaces are usually developed using a GUI development toolkit

# Evolutionary prototyping problems

## \* Management problems

- \* Existing management processes assume a waterfall model of development
- \* Specialist skills are required which may not be available in all development teams

## \* Maintenance problems

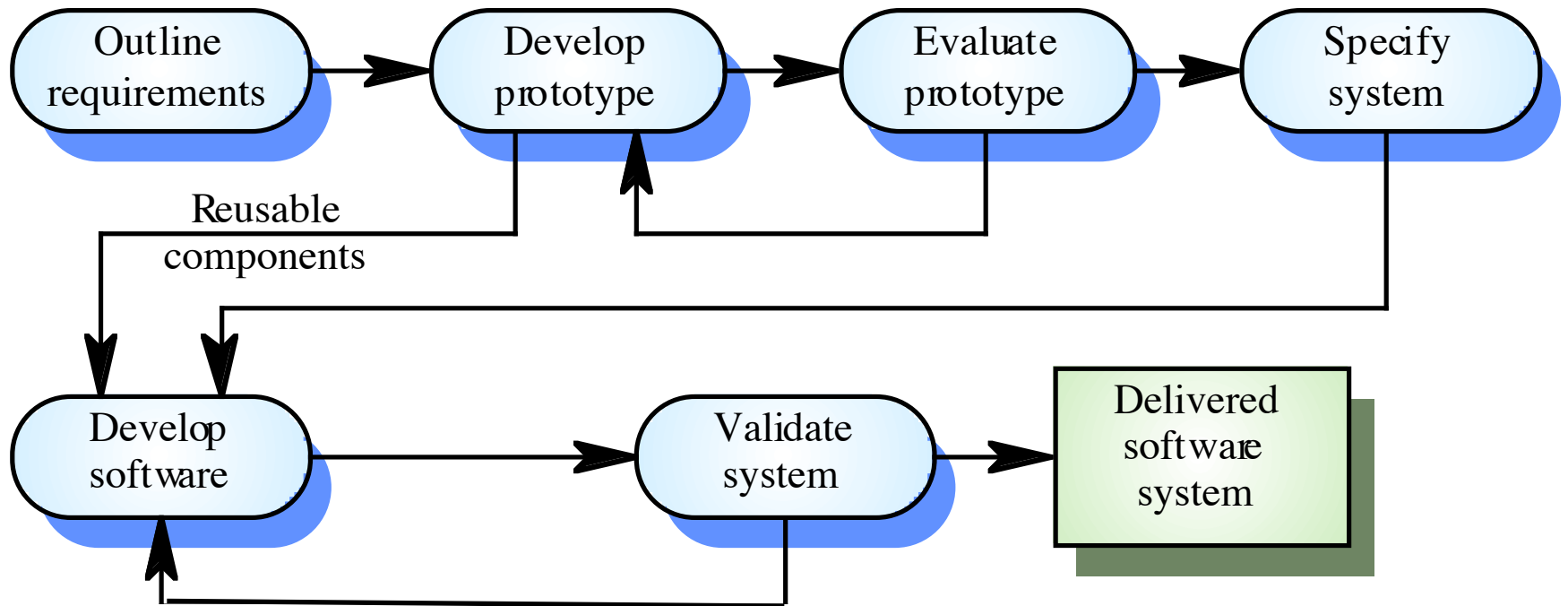
- \* Continual change tends to corrupt system structure so long-term maintenance is expensive

## \* Contractual problems

# Throw-away prototyping

- \* Used to reduce requirements risk
- \* The prototype is developed from an initial specification, delivered for experiment then discarded
- \* The throw-away prototype should NOT be considered as a final system
  - \* Some system characteristics may have been left out
  - \* There is no specification for long-term maintenance
  - \* The system will be poorly structured and difficult to maintain





# Prototype delivery

- \* Developers may be pressurised to deliver a throw-away prototype as a final system
- \* This is crazy talk (by business heads)!!
  - \* It may be impossible to tune the prototype to meet non-functional requirements
  - \* The prototype is inevitably undocumented
  - \* The system structure will be degraded through changes made during development
  - \* Normal organisational quality standards may not have been applied

# Rapid prototyping techniques

- \* Various techniques may be used for rapid development
  - \* Dynamic high-level language development
  - \* Database programming
  - \* Component and application assembly
- \* These are not exclusive techniques - they are often used together
- \* Visual programming is an inherent part of most prototype development systems

# Dynamic high-level languages

- \* Languages which include powerful data management facilities e.g. Java, Prolog, Python
- \* Need a large run-time support system. Not normally used for large system development
- \* Some languages offer excellent UI development facilities
- \* Some languages have an integrated support environment whose facilities may be used in the prototype

# Choice of prototyping language

- \* *What is the application domain of the problem?*
- \* *What user interaction is required?*
- \* *What support environment comes with the language?*
- \* Different parts of the system may be programmed in different languages. However, there may be problems with language communications

# Component and application assembly

- \* Prototypes can be created quickly from a set of reusable components plus some mechanism to 'glue' these component together
- \* The composition mechanism must include control facilities and a mechanism for component communication
- \* The system specification must take into account the availability and functionality of existing components

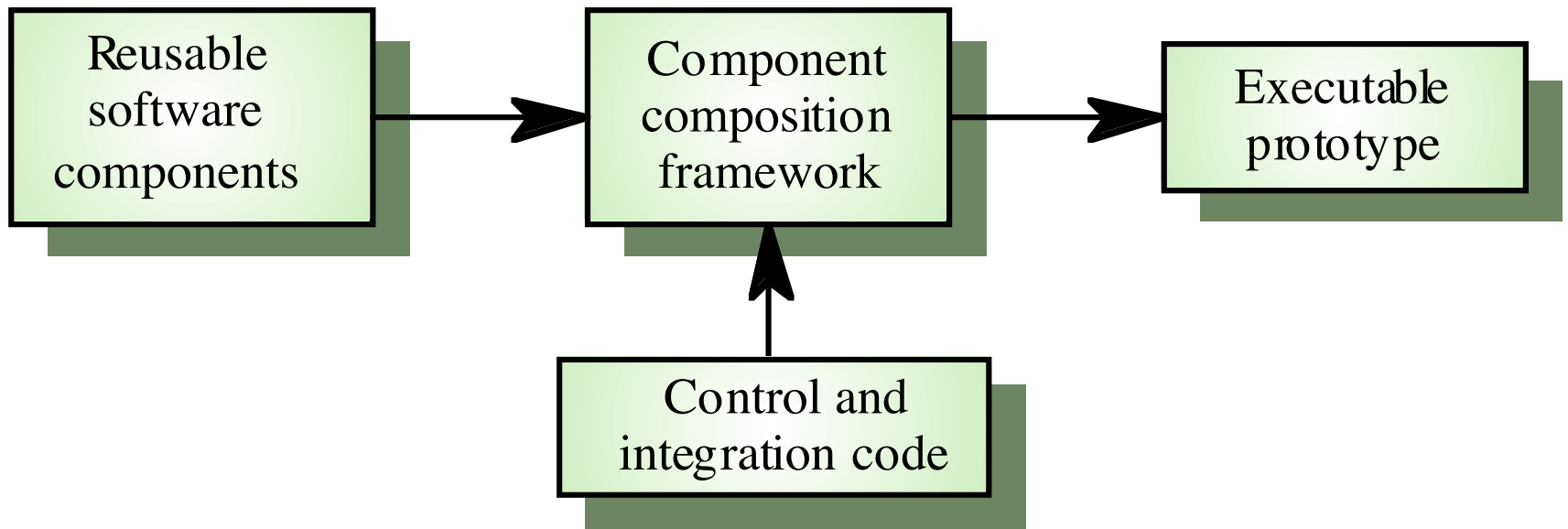
# Prototyping with reuse

## \* Application level development

- \* Entire application systems are integrated with the prototype so that their functionality can be shared
- \* For example, if text preparation is required, a standard word processor can be used

## \* Component level development

- \* Individual components are integrated within a standard framework to implement the system
- \* Framework can be a scripting language or an integration framework such as CORBA ([http://en.wikipedia.org/wiki/Common\\_Object\\_Request\\_Broker\\_Architecture](http://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture))





# Visual programming

- \* Scripting languages where the prototype is developed by creating a user interface from standard items and associating components with these items
- \* A large library of components exists to support this type of development
- \* These may be tailored to suit the specific application requirements
- \* Visual basic is actually good for this

# User interface prototyping

- \* It is impossible to pre-specify the look and feel of a user interface in an effective way. prototyping is essential
- \* UI development consumes an increasing part of overall system development costs
- \* User interface generators may be used to 'draw' the interface and simulate its functionality with components associated with interface entities
- \* Web interfaces may be prototyped using a web site editor

# The Lo-Fi Approach

- \* Mock ups of user interfaces, especially in web stuff, are often done using paper, scissors, glue and felt tip pens.
- \* Start of very rough and then refined until translated to a program like VB
- \* Might want to do this for your game!

# Key points 1

- \* A prototype can be used to give end-users a concrete impression of the system's capabilities
- \* Prototyping is becoming increasingly used for system development where rapid development is essential
- \* Throw-away prototyping is used to understand the system requirements
- \* In evolutionary prototyping, the system is developed by evolving an initial version to the final version

# Key points 2

- \* Rapid development of prototypes is essential. This may require leaving out functionality or relaxing non-functional constraints
- \* Prototyping techniques include the use of very high-level languages, database programming and prototype construction from reusable components
- \* Prototyping is essential for parts of the system such as the user interface which cannot be effectively pre-specified.
- \* Users must be involved in prototype evaluation!!!